Report No. UIUCDCS-R-77-902

UILU-ENG 77 1755
NSF-OCA-MCS73-07980-000028

IMPLEMENTATION OF AN INFORMATION-RETRIEVAL BASED CAI SYSTEM

by

Morteza Bahar

August 1977

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-77-902

IMPLEMENTATION OF AN INFORMATION-RETRIEVAL BASED CAI SYSTEM

by

Morteza Bahar

August 1977

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

*This work was supported in part by the National Science Foundation under Grant No. US NSF MCS73-07980 and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, August 1977.

ACKNOWLEDGMENT

The author would like to thank Professor David J. Kuck for his patience, encouragement and advice; Dr. Luis Osin for his guidance and many helpful discussions, and Perry Emrath for his help and cooperation.

Thanks are due to my parents for their invaluable moral support and encouragement.

The generous efforts of Cher Land and Tom Burket in the production of this thesis are greatly appreciated.

TABLE OF CONTENTS

1. INTRODUCTION

This thesis is an attempt to describe an implementation of SMITH CAI system in connection with a text retrieval system called EUREKA (thus, the name EURECAI for the entire system).

Chapter two will provide some insight for the characteristics of most CAI systems implemented to date and help to justify the design of SMITH like CAI systems.

In chapter three some characteristics of EUREKA and SMITH along with the adopted version of GUIDE (that module of SMITH that interacts with the students), are briefly discussed.

Chapter four serves as the documentation for this author's programming efforts, intending to simplify any attempt made to modify, debug and maintain the CAI system.

Chapter five is provided to guide the future author of the CAI system on how to create courses and what facilities are at his disposal. The same chapter also contains a guideline for the students, on what they can expect and how they can best benefit from the CAI courses.

Finally, a list of ideas (or potential projects) on required modification and further developments of the EURECAI is presented.

2. ON THE DESIGN OF CAI SYSTEMS

Computer-based instructional facilities have existed for over two decades now, and their use in education continues to grow. With computers becoming more cost-effective and available (ROCK75, p. 75, p. 193; STET70, p. 40), plus reports of students' interest and evidence of more rapid progress (ATKI73; HANS68; ROCK75, p. 332-335; SJOE76; LIND76) when CAI complements or replaces traditionally administered instruction, extensive growth in CAI systems design and application can be expected.

What led to writing this paper was lack of enough literature on the advantages and drawbacks of various approaches, factors involved in the design of a CAI system, and absence of conclusive recommendations in that direction.

In this paper we shall consider some factors contributing to the success of a system such as its adaptivity and ease of use. The paper also includes discussions of different teaching strategies, organization of the course contents, and their possible implications for the system characteristics.

Some more recent CAI systems whose characteristics are referenced very frequently in this paper are: SMITH (OSIN76), BIP(BARR76), COALA(GRAY77) and (the not so recent) PLATO.

In order to evaluate a CAI system we should concern ourselves with basically three areas: 1) what it offers to

the student, 2) what it demands from the authors, and 3) how reasonable an investment it is (i.e., applicability, cost, portability, etc.). With these questions in mind, a set of principles and comparison of approaches that the CAI system designer should consider are listed below. These are: adaptivity, ease of use and applicability, author-generated vs. generative CAI, lessons vs. frames, questions and answers vs. tutorial CAI, author/instructor facilities and student facilities.

2.1 Adaptivity (individualization of instruction and more)

It should be rather obvious how individualized instruction can make any mode of teaching more effective. One of the major shortcomings in teaching courses with a large number of students is lack of any personal attention to the students. Not so bright students may feel left out, lost and isolated; the bright ones may be bored; or both.

(STET70, p. 36) provides a good description of adaptivity (Stetten's "humanization", "relevance" and "individualization") and what it means to the student and to the teacher.

A considerable effort has gone into adaptive CAI systems during the recent years, examples of which are the Basic Instructional Program (BIP) at Stanford University (BARR76) MITRE's TICCET project (STET72) and the COALA project (GRAY77).

To provide adaptivity for the system, the designer uses basically "static" and "dynamic" measures. Static measures

can be in the forms of: recognizing different levels of competence for students in advance, and classifying and sequencing the learning materials according to their level of difficulty.

The SMITH[1] system serves as a good example for a system taking static measures (OSIN76). The instructor of the course may classify his students through a preliminary exam (or other means) as level-0, level-1 ..., level-k. A lower level implies more competence in the material by the student. Each frame (unit of learning material being presented) is also classified as level-0, level-1, ..., or level-k. Thus a level-i student is presented with every frame of level zero through i. A student's level may temporarily increase by his own initiative.

Stanford's 1965-1966 arithmetic project is probably the first to take static measures as discussed above. The student level would, however, change dynamically as a function of his performance (SUPP68).

SMITH also provides another level of static adaptivity measure by assigning "tags" to the frames. Each student is assigned a tag, reflecting his background, interest or school curriculum. A frame with tag X is presented only to those students having the same tag. This provides the ability to present examples or exercises that are more meaningful, motivating, or related to the student background.

---

[1] All references made to SMITH in this chapter are to the GUIDEd mode and not to the SOLO mode.

Dynamic measures are those taken depending upon the student's performance, at the time of presentation (i.e., during the terminal session). Thus, a file reflecting the student's history as he proceeds through the course must exist within the system. The CAI programs should make intelligent use of the student's history file to: a) present new materials or exercises consistent with his state of knowledge; b) remove his deficiencies demonstrated through an exercise by directing him to a review of related and helpful material (while giving him the option of skipping particular reviews), or present him with a hint or explanation to remove his doubts, and c) to control (restrict) his requests or progress when necessary. The first example of a system taking dynamic measures that we shall discuss is BIP.

BIP (BARR76), an interactive problem solving laboratory, offers tutorial assistance to the students solving introductory BASIC programming problems. The learning material is divided into tasks (which may be divided into subtasks). Tasks are programming problems of varying difficulty, identified by their text, skills and model solution. Task selection is based on both: 1) the Curriculum Information Network which describes the problems in terms of skills, and 2) a model of student's knowledge (i.e., student's history file). The MUST set holds a review list of skills still not mastered by the student. This set aids the task selection procedure.

COmputer-Assisted Learning Applications (COALA) teaches introductory electrical engineering network theory and was

developed at the Educational Technology Unit of the Victoria Institute of Colleges (Australia).

COALA (GRAY77) can present a concept in four different modes: as a rule, by example and counter-example, drill and practice problem, or by application. Two factors considered in the frame selection are: 1) A pace rating determined by the student's performance in the previous frames, 2) A most suitable presentation mode determined by the student's previous responses. Since the pace rating can also be a negative integer, it makes stepping back a few blocks of frames possible. This may be considered an instance of a system-initiated review facility discussed earlier.

Upon an incorrect response to an exercise, SMITH provides helpful material (covering topics predetermined by author) to remove the student's specific deficiency in the related topics. The nature of the review is a function of the number of his earlier failures in the same exercise, and the particular incorrect answer chosen. The system reply to the student's incorrect response may also be a brief explanation or comment as specified by the author. Note that the student at any point in the review mode can end the review and return to the corresponding exercise. This feature is a very desirable one since often the student identifies his error very shortly after he has made it and then a "forced" review is not an appropriate approach.

It is hard to make specific comments on the PLATO project, implemented at the University of Illinois, since every lesson

on PLATO could be written by a different author and based on his teaching strategy and approach. To partially satisfy the equal time advocates, however, we shall refer to PLATO, but only to those lessons included in the introductory computer science courses (NIVE76). The students taking these courses are treated in about the same way. Some lessons may offer an option to skip frames within the lesson, but that is hardly enough and it is rather risky to skip the frames of possibly unknown content. For example, a student may know about one form of IF statement in FORTRAN, but not about all its forms, thus by using the skip option he may miss some useful frames.

It is unfortunate to see that most CAI systems have been designed with a homogenous student population intellect, competence, and performance in mind. Easy Assembler SYstem (SJOE76) and COMPUTER-TUTOR (LIND76) are recent examples of the many such CAI systems.

The reader should note that generative CAI systems are inherently more adaptive than the author-generated systems (both are discussed further in this chapter). This follows because a generative CAI system continuously maintains a model of the student's knowledge and is designed to provide information in precisely the area needed the most. Furthermore, all decisions are made dynamically and depend on the student's performance.

2.2 Ease of Use & Applicability

This criterion, this author believes, is an important

one, since it distinguishes between an experimental CAI system and a practical one. It includes factors such as availability, flexibility, cost, portability, ease of implementation and time-space requirements. Designs locked into providing instruction on a specific material; implemented in an assembly language; requiring a fixed hardware configuration or special equipments; ... are inflexible to use for teaching other materials and skills, and costly (if possible) to implement on different hardware facilities.

A major characteristic shared by most CAI systems is that they are locked into teaching (or testing) a specific subject (e.g., a programming language) or skill (e.g., programming, problem solving).

Computer-Aided Flow Diagram Teaching System (KOFF76), BASIC Instructional Program (BARR76), Easy Assembler SYstem (SJOE76) and the SOPHIE system (BROW74) are recent examples of such systems.

This application dependence of such systems is probably their major disadvantage. If often implies that the author of the course and the system designers should work as a team, a team which may not be effective due to the diversity in the members' background.

A very good counter example for such systems is SMITH. This system does not offer a specific CAI course, but rather application independent facilities which may be used to create courses to teach any subject, with no programming requirements imposed upon the author!

A desirable feature of CAI programs is their portability (whether they are application independent or not). COALA (implemented in BASIC) and SMITH (implemented in PL/1) were designed with this criterion in mind. PLATO lessons implemented in the TUTOR language and requiring special screen and keyboard (as does TICCIT) serve as counter-examples.

Evaluation of CAI systems in terms of cost and time-space requirements, with respect to their capabilities, characteristics and effectiveness is still an open research subject.

2.3 Author-generated (address-oriented)[2] vs. Generative (information-oriented) CAI

The distinction between these two approaches to CAI system designs is important, mainly because of its possible implications on ease of creating a course and quality of instruction. A good understanding of these two approaches and characteristics of those CAI systems based on each, should be very useful, if not essential to the CAI system designer(s). This understanding aids the characterization of the initial design approach.

Author-generated CAI is the most common style. The author decides on the course contents and goals, organizes the material into frames of text and/or exercises, specifies the sequence of presentation and all the branching decisions.

The major advantage of this approach is that the frames can be well-written and so organized to serve as a great

[2] The terms in parenthesis were suggested in (OSIN76).

motivating factor to the student. This is an important ingredient missing from most CAI systems; questions and answers systems, in particular. (HICK74) lists the following research recommendation: "research into techniques for maintaining the increasing motivation which can be incorporated into the instructional strategy".

Another advantage of the author-generated CAI approach for the creators of the courses rests with the smaller skill and programming requirements. This will be more obvious when the generative CAI approach is discussed. Note, however, that in both strategies, the amount of time and skill required from the author is still excessive.

The most important disadvantage of the author-generated CAI approach is its inherent lack of adaptivity. The system performance is always limited by the author's ability to predict. There exists the implied assumption that the material is well-understood and structured, and the student is well-characterized -- an inflexible and unreal assumption, in general. Furthermore, to provide adaptivity by specifying branches as a function of student's history and competence can become a very complex task.

Introductory computer science lessons on PLATO are written with this approach in mind. A lesson consists of a pre-determined (pre-programmed is more accurate) sequence of frames, with a set of predicted answers for exercises (BAHA76).

In COALA, the author may specify the order of presentation of frames within a module, and the answers to exercises are

also pre-specified by him. COMPUTER-TUTOR presentation is based on a simple sequence with no branching.

There are quite a few course-author languages (ZINN71) that allow the author to create a course based on an author-generated CAI strategy (e.g., IBM's COURSEWRITER, Course Writing Facilities of Hewlett-Packard).

In the generative CAI method, the system produces a frame of learning material or problem, analyzes the student's response and takes the appropriate action following his response or request. Therefore, the system has all the necessary "knowledge" about the course being taught. The author must provide algorithms to produce suitable questions and answers, directories of equivalent terms and expressions, plus all the facts, dependencies and interrelations. In short, the author's knowledge of the subject and more should be programmed into the system -- certainly a non-trivial task! Semantic networks are often used to represent the system knowledge.

An advantage here, is that the course can provide information in the area most needed by the student. This is due to the fact that all decisions are made dynamically based on the student's performance and state of knowledge, rather than a pre-determined sequence of material. Note that this also means maintaining and representing the student's knowledge as he progresses through the course.

An important disadvantage of this approach is the enormous amount of effort required by the author, and the complex nature

of representing information, student's knowledge, and the response analysis. Furthermore, since facts, relations and algorithms do change from one context or course to another, systems based on this approach are locked into the subject being taught, as discussed under "Ease of Use and Applicability".

Finally, even though the system may intelligently diagnose the learner's difficulties, it is not always possible to find the material required to remove student's deficiencies in the data-base. In BIP, the system may determine the skills the student has yet to master, but there may be no suitable task developing or testing these skills, i.e., "a hole in the curriculum" (BARR76).

The instructional program of BIP, maintains his "knowledge" of the course, through access to the Curriculum Information Network which consists of 100 programming problems and the skills developed in solving each.

SOPHIE system (BROW74) has a great deal of knowledge about what it teaches -- troubleshooting a complicated electronic circuit. SCHOLAR (CARB70) is a classic example of information-oriented approach. Computer-Aided Flow Diagram Teaching System is considered "knowledgeable", and it can question the student on his flowchart. There are no pre-stored questions and problems.

As the reader might have guessed, a suitable combination of both approaches could entail some advantages and soften the disadvantages of each strategy. SMITH's design is based on this scheme. The author of the course provides information

about each frame of material, by identifying the topics covered in (or related to) that frame. Answers to the exercises are predicted by the author, and for the incorrect answers a list of topics (not frames) to be reviewed are specified by him. The author may require some ordering among the frames, but at the time of the presentation the sequence of frames is pre-determined.

The advantage of SMITH's approach is the more reasonable amount of work it requires from the author. In address-oriented CAI, the author must specify all the branches, and in generative CAI, he has to provide all the facts, but here, he specifies a small subset of both. For specific figures on the author time requirements to create a course see (OSIN76) and chapter 5.

2.4 Lessons (modules, tasks) vs. Frames

It has been quite common for CAI system designers to recognize a logical breakdown (or hierarchy) of sets of materials making up the CAI course. This logical observation has often been mapped or programmed into the physical/actual implementation. In other words, the content of the course has been structured into some representation or model, one which is, preferably, easily manipulatable and accurate. The purpose of this section is to point out how the course representation may affect system-student interactions. Let us now cause no further confusion by resorting to some examples.

In BIP, the goal is to enable the student to master a set of techniques. Each technique consists of a set of skills. A task is a problem designed to test or develop a set of skills. A skill may be mastered through more than one technique and is used or developed by several tasks. Each task has a set of subtasks and hints among other components.

This view is very closely programmed into the system (e.g., it is essential to tc the task selection process). The frames[3] and tasks carry no identification from the student's view, so he cannot request a particular task. A given task is presented only when the system considers it appropriate according to the student's needs and the pre-specified sequence of techniques.

In SMITH, the course is organized according to the topics to be taught. A given frame exists because of its relation to one or more of the topics covered in the course. Therefore, when organizing the course, the frames are sequenced as a function of the order by which the topics are discussed. No hierarchy or grouping of the frames exists in the (representation of the) curriculum, as it does in PLATO and COALA. Between the frames and the topics, however, we can observe a many-to-many relationship (a complex plex structure), if we wish to.

In PLATO, the material covered in the course is structured like a text book. Each lesson corresponds to a chapter and

[3] Let's define a frame as a unit of course material text being presented to the student, with a response expected from him after the presentation. The problem description, subtask text or hints are all examples of frames in BIP.

can be identified similarly. Each lesson is divided into different sections which are subordinate to the lesson and accessible by the student only through the lesson. The frames within a section are presented sequentially (some lessons may offer a skip option to allow "faster" progress) (BAHA76).

COALA modules are similar to PLATO lessons, and cells to sections, but the frames within a cell are not presented in a similar fashion, as discussed under "adaptivity". Frames are created through the FRame Entering System for COALA (FRESCO), which generates one program per frame.

Let's now discuss some of the implications and characteristics of various organizations of curriculum upon the mode of system-student interaction.

In BIP, if the student has once "mastered" a skill, and now wishes to review that skill, he has no direct control over choosing an appropriate task.

In SMITH, if the concept the student wishes to review has been explicitly identified by the author as a topic, or discussed under several known topics, he has direct access to the corresponding frames. Otherwise, he is forced to "navigate" through a subset of the frames he has already seen (i.e., he has to request to review every topic, which allows him to see only those frames that are most relevant to a given topic).

Under PLATO, and to some degree COALA, the student may have direct access to a section (not the particular frames), but only if he can identify the lesson and the section

precisely. If the concept of interest is scattered throughout various lessons, however, he has no choice but to navigate through all the suspected lessons!

Sometimes, a student may wish to review a given example or explanation or re-do an exercise. BIP provides no such student control. In SMITH, as pointed out earlier, only a subset of frames can be accessed through review requests. The student may, however, step back through the sequence of frames he has already seen. No solved exercise may be re-attempted. A PLATO student may re-view any frame he wishes, if he knows where to find it. Re-view of a given frame seems unlikely for a COALA student, due to the random and history-dependent frame selection within a cell.

It may be desirable if the student could continue his instruction from exactly where he left off during his previous session. It seems CAI systems with curricula organized as a set of lessons or tasks could not provide such a feature. A PLATO student after solving a few exercises and "reading" a few frames within a lesson, will be re-presented with them, if he exits the lesson and returns later (even if within the same session). BIP student's return point is not within a task but after the last finished task. Only in SMITH, where there is no hierarchy imposed upon the frames, the student may resume with the current frame at his time of departure.

A final observation should be made, about the actual implementation of a lesson-oriented system (e.g., PLATO) vs. a frame-oriented approach (e.g., SMITH). Once a PLATO student

requests a given lesson, the program and the lesson text are entirely loaded into the memory (extended core storage). More than one student may share the same lesson, but if they choose different lessons, a great amount of storage should be made available to them. In the frame-oriented approach since each frame can independently exist, it can be brought into the memory and presented to the student only when needed. In SMITH, the text is totally separate from the programs, and as a result the student has access to enormous amounts of material at no extra main memory requirement (this approach is considered information-retrieval oriented). A SMITH course can offer the student a variety of introductory, complementary and essential material plus exercises, on a given topic, as opposed to PLATO lessons, where the amount of text is of direct (memory) concern.

It is not obvious, in general, which approach to curriculum organization is a superior one, even though in the above discussion, the frame-oriented approach does seem more flexible than the lesson-oriented structures. But, let's note that, whatever type of curriculum representation a designer chooses, it should be with a given teaching strategy and the course-content-related objective in mind, and not purely as means of facilitating the implementation (an advantage of PLATO course organization is that writing the lessons may be divided among several authors). It should also allow a reasonable manipulation of the material by the

student (in forms of reviews, repeats, etc.) as well as the flexibility to grow in options and in the amount of text.

This area does still require more research. In 1974, Hickey listed: "research on the organization of content for easy retrieval by students who are utilizing a learning strategy" (HICK74) as an important research topic.

2.5 Questions and Answers vs. Tutorial CAI

Since, in the field of CAI there exists no universal agreement on the meaning of most commonly used terms, let us first define those we have already seen in the section title.

Questions and Answers CAI systems are those offering student-controlled instruction. The student may ask to be presented with some information, explanation and examples; or to be tested on a given topic or skill. Some of these systems carry on an English-like conversation with the student and most have little or no control and teaching strategy (e.g., GEO-SCHOLAR (CARB73, COLL73)).

Drill and practice, problem solving and dialogue systems are all treated as subsets of questions and answers systems in this discussion. The drill and practice system reinforces learning of a concept the student has already been exposed to, by means of providing examples and/or exercises on that concept. Some PLATO lessons (e.g., chemistry lessons (BITZ72)), the CLOSE program ((ROCK75, appendix A), and EASY (SJOE76) are examples of drill and practice systems.

The reader is referred to the description of problem-solving (e.g., BIP) and dialogue systems (e.g., SOCRATIC (FEUR65)) found in (ROCK75).

Tutorial systems are by far the most widely implemented CAI systems. Their primary purpose is to teach the students about new facts and concepts. This is done through: 1) Introduction of a new concept (or motivating the learner); 2) Description of the concept, presentation of facts; 3) Clarification (by examples or supplementary material); 4) Exercises and problems.

Some PLATO lessons (e.g., introductory computer science courses (NIVE76)), SMITH (OSIN76), COMPUTER-TUTOR (LIND76) are examples of tutorial systems.

The distinction made here, between tutorial and questions and answers CAI systems should be rather obvious, since they have different objectives. The primary focus of the former is to introduce new facts or concepts, and provide some examples and exercises, but the latter is designed to re-enforce what the student has previously studied, by offering him examples and mostly exercises.

Most dialogue systems (e.g., GEO-SCHOLAR) are able not only to generate (or present) questions but answer the student's question, as well. But due to some of the following shortcomings they are considered as questions and answers CAI rather than tutorial: 1) They lack control and (teaching) strategy. 2) To teach some subject domain, a stand-alone, self-complementing CAI system of this type is insufficient (GRIG74).

3) Organization of the course content in a student-controlled environment, the student's strategy in learning (in that environment), and his estimate of his own competence are all topics that (may still) require more research (HICK74). 4) As discussed earlier, question and answer dialogues may not be motivating, and these systems are often application dependent.

In short, questions and answers CAI systems are appropriate to use for only a part of the instruction process, as opposed to tutorial CAI which can be stand-alone and self-complementing. The significant purpose that drill and practice, and problem-solving CAI systems can serve, however, should not be underestimated. Availability of these systems has been very helpful in teaching some topics and skills such as accounting, programming, problem-solving and the like.

The major weakness of tutorial CAI, simple tutorial in particular, is that it is often author-generated, forcing the inherent limitations as discussed under previous sections.

Once again, a more reasonable design would be one based on a mixed approach, one which is tutorial in character but provides examples and exercises as evenly or effectively as it presents facts and concepts. This would also imply transfering more control to the student (from the author) so that he may practice and exercise the concepts according to his needs and interests.

2.6 Author/Instructor Facilities

Since there normally has not been much concern for a

course author's workload in CAI system design, few non-CAI researchers have been attracted to CAI. The major burden for authors is that often they must take part in the design of the CAI system rather than only dealing with curriculum development for that system. As a result, an enormous amount of time and skill have been required of authors in the past (The reader is referred to (OSIN76) and the following references listed in that paper: FROM73, ROSE72, DONI71, HAEF71).

As discussed earlier, due to the application dependency of most CAI systems, the participation of the teachers in their design and implementation should be no surprise!

The designers should think of the potential authors by answering these questions:

1. Who can be an author for the system? Competence in what skills are required of an author?
2. How easily can he create a course? Is it facilitated by other software support?
3. Are there facilities to change or update the course contents (without having to recreate the course)?
4. Are there facilities to monitor or record a student's performance, attendance, difficulties, communications, etc.?

COALA aids the author in creating frames. He must, however:

"1) draft a screen layout, and organize any branching which is dependent upon the student responses within the learning materials;

2) Code the screen layout (and branching) into appropriate

frame entering instructions; and

3) enter the instruction into the system."[4]

(GRAY77, p. 75)

On-line facilities also exist for preparation of diagrams used within a frame.

One of the functions of the COALA supervisory system is to monitor and record student access as well as record/ report his performance in the course.

Among the TICCET aims was providing a variety of features to facilitate author's maintenance and improvement of courses. They include features to: identify weaknesses in course contents, record student progress, and detect student problems (and alarm the instructors).

PLATO authors must learn TUTOR to create a lesson, but the instructional facilities provided on the system are excellent. Instructors may collect various statistics on students such as their performance, attendance and progress history. The instructors can easily communicate (in real time) with one or more of the students. This is especially useful when the instructor and student(s) work at different sites. Instructors, authors and the students may also leave messages (e.g., remarks on a lesson) for each other (NIVE76, BAHA76).

To change a lesson on PLATO means one must modify the program of that lesson. Due to complexities of graphic techniques involved, most lessons are under continuous revision, as the programmers (i.e., <u>authors</u>) gain more experience.

---

[4] Author's emphasis

Creating a course of lessons is thus considered very time consuming and/or labor intensive.

SMITH provides a variety of software supports to aid the preparation of the course. These programs can detect weaknesses ("holes") in the curriculum, inconsistencies in the specification, or the author-generated sequence. The support is available to organize the course using the author's initial specification while requiring no further intervention. The aim in the design of SMITH was to require no programming or special skills other than the author's ability to teach the subject (i.e., decide on the topics to be covered by the course, organize frames of material, exercises, and answers, related to those topics).

2.7 Student Facilities

Since in any CAI system the main concern is the satisfaction and learning of the student, almost everything we have discussed is basically related to what the system can offer to the student. In this section, however, let us consider some other desirable features which can be included in the system design or the final implementation.

The first question is how well the student and the system communicate or interact. Systems which allow a natural-language-like conversation are probably the most desirable and natural for the student, but the implementation of these is still experimental, and can be too complex to be practical.

The communication language should be simple to learn and resemble natural and human-like conversation as much as possible.

COMPUTER-TUTOR serves as a good example. Its student commands are: QUIZ, LESSON, PROGRAM, REVIEW, INFORMATION, OUTLINE, QUESTION, BYE.

BIP makes many instructions available to the student. In the author's opinion, however, students with no programming experience (who BIP aims at), will first have to be taught (by means of explanations and examples) what some of these commands mean and when they should be used, before they are expected at the terminal. Some of the BIP student commands are: HARDCOPY, TASK, SUBTASK, SCRATCH, FILES, SAVE, GET, MERGE, KILL, etc.

The most complex and important portion of interaction with the student is response analysis (answers to exercises, in particular). A reasonable system must check for spelling errors, for an undetected spelling error can totally confuse the student and lead to a mistrust of the system. A list of synonymous words or expressions should be maintained (in a thesaurus). In mathematical problems, equivalent expressions and values in different units or within a particular tolerance should be acceptable. Partially correct answers should be so indicated in error messages (e.g., if the answer is in two words, and the student has one word right). Error messages should be clear, helpful and understandable.

In this direction, COALA provides one of the better facilities. Student response evaluation programs: take into account alternative responses, accept mathematically equivalent results, correct misspelling (and indicate it), maintain a file of synonyms, etc. (GRAY77).

The second point of interest is the student's interactive role. What is expected from him? It is probably undesirable to require the user to continuously make decisions on what to see (now), to skip, or to see later. The student does not always know what he does not know, or what he should see. In short, the system should have an author-designed pattern to follow, yet allow student-initiated changes in the presentation. The student should be able to get around restrictions imposed upon him or the curriculum, if he finds it necessary. For example, if he chooses to or is forced into a review, he should not be required to re-do the exercises he has already successfully completed; or if he has failed an exercise several times he should be allowed to proceed or request the answer.

Some exercises on PLATO computer science lessons do not allow the student to proceed if his answer is incorrect and the system's reaction in this case is a simple "no"! This intolerable situation often cannot be resolved without the instructor's assistance (BAHA76).

The BIP supervisory system requires the student's self-evaluation following each task, weighting the results in the selection of tasks.

In SMITH, a wrong response to an exercise triggers an explanation or re-presentation of the frames (exercises are excepted) most related to the student's deficiency. The student may, however, end this review and return to the exercise at any point during the review.

The final point to be discussed is the "extras" from the student's point of view. Features like graphic capabilities can result in very useful illustrations, introduce a variety in mode of presentation, attract student's attention, etc.

PLATO provides probably one of the best graphic facilities for general purpose CAI today. Graphic features are also available on: BIP, COALA, TICCET, and others.

Instead of attempting to classify such "extras", let us settle for a few instances of efforts in this direction. The COALA system enables the student to search through a data-base of material relevant to the course. He may also request definition of a given word in terms of explanatory text or illustration (graph). COMPUTER-TUTOR can easily be used to give a placement test.

One of the recent implementations of SMITH (called EURECAI) provides full access to the data-base by means of a reasonable query language. This is discussed in the following chapters.

The BIP system provides probably one of the best sets of debugging and problem solving aids among the systems of its kind. This includes features: to trace the execution of the program; to observe the output of a sample solution; to pinpoint

the student's error (i.e., an interpreter), to provide a sample solution, hints and help among others.

LIST OF REFERENCES

ATKI73 Atkinson, R.C., J.D. Fletcher, E.J. Lindsay, J.D. Campbell, & A. Barr, "Computer-Assisted Instruction in Initial Readings: Individualized Instruction Based on Optimization Procedures," Educational Technology, 13, 17, 1973.

BAHA76 Bahar, Mory, Personal observation and experience: The author was a teaching assistnat for C.S. introductory courses during spring and fall of 1976. Duties also required being the instructor of several PLATO sections, 1976.

BARR76 Barr, A., M. Beard, & R.C. Atkinson, "The Computer as a Tutorial Laboratory: The Standard BIP Project," International Journal of Man-Machine Studies, 8, 576-596, 1976.

BITZ72 Bitzer, D.L., B.A. Sherwood, P. Tenczar, "Computer-Based Science Education," presented at the conference on "Utilization of Educational Technology in the Improvement of Science Education," UNESCO, Paris, September 1972.

BROW74 Brown, J.S., R.R. Burgon, & A. Bell, "An Intelligent CAI System that Reasons and Understands," (BBN Report 2790), Cambridge, Massachusetts: Bolt, Beranek and Newman, 1974.

CARB70 Carbonell, J.R., "AI in CAI: An Artificial-Intelligence Approach to Computer-Assisted Instruction," IEEE Transactions on Man-Machine Systems, December 1970.

CARB73 Carbonell, J.R. & A.M. Collins, "Natural Semantics in Artificial Intelligence," Proceedings of the Third International Joint Conference on Artificial Intelligence, Stanford, California, August 1973.

COLL73 Collins, A.M., JJ. Passafiume, L. Gould, & J.R. Carbonell, "Improving Interactive Capabilities in Computer-Assisted Instruction," (BBN Report 2631), Cambridge, Massachusetts: Bolt, Beranek and Newman, 1973.

DONI71 Donis, J., "Computers in Education: Present Situation and Development Trends," in A. Daniels (Ed.), Educational Yearbook, 1971-1972, London: The British Computer Society, 1971.

FEUR65 Feurzeig, W., "Towards More Versatile Teaching Machines," Computers and Automation, March 1965.

FROM73 Fromer, R., "Author Support Requirements of a Computer-Based Instructional System," Educational Technology, 13, 18, 1973.

GRAY77 Gray, D.C., J.P. Hulskamp, J.H.S. Kumm, S. Lichtenstein, & N.E. Nimmervoll, "COALA - A Minicomputer CAI System," IEEE Transactions on Education, E-20, February 1977.

GRIG74 Grignetti, M.C., L. Gould, C.L. Hausmann, A.G. Bell, G. Harris, & J. Passafiume, "Mixed-Initiation Tutorial Systems to Aid Users of the On-Line System," (NLS) (ESD-TR-75-58), Bedford, Massachusetts: Deputy for Command and Management Systems, Electronic Systems Divisions, in A. Barr, M. Beard, & R.C. Atkinson, "Computer-Based Tutorial Laboratory: The Standard BIP Project," International Journal of Man-Machine Studies, 8, 576-596, 1976.

HAEF71 Haefner, K., "Computer-Assisted Instruction at University Level in Science and Medicine," Naturwissenschaftliche Rundschau, 24, 12, 1971.

HANS68 Hansen, D.N., W. Dick, & H.T. Lippert, "Research and Implementation of a Collegiate Instruction of Physics Via Computer-Assisted Instruction," Technical Report No. 3, Florida State University, Tallahassee, Florida, November 1968 in John Fralich Rockart and M.S.S Morton, Computers and the Learning Process in Higher Education, New York: McGraw-Hill Book Company, 1975.

ROSE72 Rosenbaum, P., "Toward the Automated Workbook: A New Direction for CAI," Educational Technology, 12, 36, 1972.

SJOE76 Sjoerdsma, T., "An Interaction Pseudo-Assembler for Introductory Computer Science," ACM Joint Bulletin SIGCUE-SIGCSE, February 1976.

STET70 Stetten, K.J., "The Technology of Small Local Facilities for Instructional Use," Mitre Report No. M69-39, Mitre Corporation, Bedford, Massachusetts, June 1972.

SUPP68 Suppes, P., M. Jerman, & D. Brian, "Computer-Assisted Instruction: Stanford's 1965-1966 Arithmatic Program," New York: Academic Press, 1968.

ZINN71 Zinn, K.L., "Requirements for Programming in Computer-Based Instruction Systems," in A. Daniels (Ed.), Educational Yearbook, 1971-1972, London: The British Computer Society, 1971.

3. A "BETTER" CAI SYSTEM

In this chapter, the reader will find a brief description of EUREKA (an information retrieval system), SMITH (a computer-assisted instruction system) and the system obtained by linking an adopted version of SMITH to EUREKA (called EURECAI). Their inclusion in this thesis is primarily for the sake of completeness. The interested reader is referred to the literature provided by those originally involved in the design, as well as the implementation of these systems.

(OSIN74) provides the detailed description of the background, approach, design, implementation and results for SMITH. (OSIN76) is a recent report (and a better one to read) on the SMITH system and includes discussion on the developments of a newer version called SMITH II, implemented at the University of Michigan.

(STEL74) provides some background to the EUREKA system. The description of the system and also a user manual for EUREKA are provided in (MORG76). Some evaluation and development related to EUREKA are documented in the following reports: (RINE76), (MORT76). Since most EUREKA modules are under continuous revision, none of the above reports are considered up to date.

3.1 Introduction to SMITH

The SMITH system is an information retrieval oriented computer-assisted instruction system. It was designed and

developed by Luis Osin (OSIN74, OSIN76) at the Israel Institute of Technology. Most of the material in this section is taken from (OSIN76).

As pointed out in chapter two of this thesis, SMITH is a tutorial CAI system, with some adaptivity features, flexible in usage, frame-oriented, and requiring no programming by the authors of CAI courses. Its design is based on an intermediate strategy between the two extremes of author-generated and generative CAI. The author specifies some information on the frames and the contents of the course, and some on the branching and remedial loops. With intelligent use of this minimal information SMITH's teaching strategy can provide a suitable and adaptive presentation to the students.

Let's now briefly discuss the nature of the author's specification involved in the design of a SMITH supported course.

3.1.1 The Author Specifications

3.1.1.1 Specification of the topics. A list of topics to be covered in the course or required as prerequisite to it are provided. A topic is identified by a topic number.

3.1.1.2 Indexing of frames. Each frame is indexed by a set of couples. Each couple consists of a topic number and a qualifier describing the relation between the frame and the topic. The list of possible qualifiers is given in Table 1.

Table 1

Qualifier	Function
I	(Introduction) The frame introduces the topic.
N	(Necessary) The frame presents material essential to learning that topic.
C	(Complement) The frame serves to clarify (by example or explanation) the topic.
E	(Exercise)
P	(Problem)
R	(Required) Knowledge of the topic is required for learning other topics covered in the frame.

3.1.1.3 Sequencing specifications

(a) Frame precedence relations (FP)

Here the author may specify some restrictions on the final sequence of the frame presentation. The specifications in this part are in the form of pairs, i.e., the first frame is to precede the second.

(b) Chains

All the frames specified as a chain are entered into the

sequence with no insertions allowed within the chain.

3.1.1.4 Instructional specifications

The presentation is not only based on the sequence of the frames, but also adaptive to the student's level of competence and background.

(a) Levels

The author must assign a level to every student and every frame in the course. A lower level, for the student, implies more competence in the material, and for a frame implies a more difficult and essential frame for covering the course. A level-i student is presented with all frames of level-i or lower.

(b) Tags

Each student is assigned a tag, reflecting his background, interest or simply school curriculum. Similarly the frames of interest are assigned a tag. This allows two students with different backgrounds to be presented with examples or exercises relevant to their own background and interest. Therefore, a tagged frame is presented only to those students having the same tag. Un-tagged frames are presented to all.

3.1.1.5 Specification of special frames

There are two types of special frames:

(a) Extensions

These frames are extensions to the main frame, with the purpose: 1) to give more explanation of the contents of the main frame, or 2) to mention topics which may be of interest

to a minority of students, or 3) to give a hint towards the solution of an exercise or problem.

(b) Exercises

For each exercise the author specifies a list of predicted answers. For each (or a group of) incorrect predicted answers he provides a remedial group. An incorrect response by the student is an indication of his lack of understanding of particular topics. These topics should be placed in the remedial group.

This concludes the author's effort towards creating a new course.

3.1.2 Sequence

The next step is to order the frames satisfying the author's sequencing specifications as well as the following criteria:

(a) Topics already based on student's previous knowledge should be presented first.

(b) New topics should be presented in a smooth and continuous manner (in successive frames).

(c) Presentation of a topic should be in the form of a set of introductory (I) frames, followed by necessary (N) frames, supplemented by clarification (C) and exercises (E) or problems (P).

This step is performed automatically, using the sequencing program called SEQUENCE.

3.1.3 The Teaching System

Each SMITH student is assigned a level through a test or interview, and has an activity file containing his attributes and history of progress in the course.

The student may interact with SMITH in two different modes: guided (author-controlled) and exploratory (student-controlled). GUIDE and SOLO are the programs corresponding to these two modes.

There are two types of frames presented to the student -- expository and exercise.

(a) Expository frames

Any frame which is not an exercise is expository. The student's response to such frames can be any of those listed in Table 2.

Table 2

Character	Meaning	System Action
U	Understood	Present the next frame in the current sequence.
C	Clarify	Presents new frames clarifying the transition between the previous frame and the current one.
B	Back	Presents the previous frame.

S	Suspend	Terminates the session.
?	Doubts about certain topics	Allows the student to review the topics of interest.
R	Resume	Returns to the original sequence -- the sequence he was following when he responded with "B", "C", or "?".

(b) Exercises

The student's response can be either an answer to the exercise or any of the commands listed in Table 2, with the exception of "U". If the student fails the exercise, he will be presented with a review of the concepts (i.e., topics) he has failed to learn. If this is the first time the student has failed the exercise, the student is presented with only those frames which are essential to learning the corresponding topics and are to precede that exercise (the current frame). Recall that this precedence relationship is specified by the author (see section 3.1.1.3). Subsequent failures will result in more intensive reviews (i.e., more frames).

3.1.4 System Implementation

The SMITH characteristics that justify the linkage between SMITH and EUREKA are outlined in this section.

There are basically four modules required by the instructional process:

(a) The teaching program (GUIDE or SOLO).

(b) The course description file.

(c) The frames of material (i.e., the course content).

(d) The student activity or history file.

In the teaching process, GUIDE (for example), uses the information in files (b) and (d) to select a frame from file (c). Each frame in file (c) is uniquely identified by a key (the frame library number). The file (c) is accessed through an indexed sequential access method. During the instructional process, for every student-terminal, the teaching program (a) and the files (b) and (d) are loaded into the memory. This is obviously an impractical approach, unnecessary, costly (in terms of memory) and may degrade the system performance (i.e., poor response time, excessive memory management overhead, etc.). As (OSIN76) suggests too, only one copy of (a) and (b) should be loaded for all students.

3.2 Introduction to EUREKA

EUREKA is a mini-computer based experimental information retrieval system, retrieving text under a multiprocess, multiuser monitor. A query language is provided which allows the terminal user to perform rapid searches through the data-base. The data-base consists of a set of documents, each indexed by every word occurrence in the document. It is this inverted file organization that allows retrieval of a set of documents satisfying a query within a reasonable amount of time.

Facilities are also provided to view (or print) sets of documents (satisfying a user query), perform full text search of the documents, and maintain the result of previous searches performed (i.e., a user file).

The points of interest to us concerning EUREKA and its implementation are: 1) The multiuser executive system (described in MILN75). 2) The text retrieval mechanisms and 3) The ability to store and maintain a moderate amount of text in the data-base.

3.3 Introducing EURECAI, the Dynamic Dual!

During the fall of 1975, we decided to implement GUIDE, the interactive instructional program of SMITH (author-controlled CAI program) linked with the EUREKA system.

The advantages were obvious. To the author it meant: 1) A previously created library of frames (i.e., the text in EUREKA data-base), 2) ease of manipulation of the text frames (i.e., each frame is easily identified by document and paragraph number) and 3) search facilities to provide access to all the frames of interest in the data-base (i.e., shorter course preparation time).

To the student, it means access to EUREKA data-base which should normally contain the texts related to the course -- recall that in SMITH, frames being utilized in the course are only a subset of a library of frames, previously available only to the author.

At last, and certainly not the least, are the advantages from the system and implementation point of view: 1) Only one copy of GUIDE is kept in the memory, regardless of the number of students, 2) the ability to keep only one copy of the course description file in core (not actually implemented at the present time) and 3) retrieval of frames is left to EUREKA. This is actually a factor that degrades system performance. EUREKA performs a sequential search through a document to access a given paragraph, thus requiring a considerable CPU and I/O time. In SMITH, a frame is accessed through an indexed sequential method which may require no more than two disk accesses (assuming the index tables are not kept in core).

LIST OF REFERENCES

MILN75 Milner, J.M., "A Multiprocess, Multiuser Executive for an Experimental Information Retrieval System," Report No. UIUCDCS-R-75-736, M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, Illinois, August 1975.

MORG76 Morgan, J.K., "Description of an Experimental On-Line, Minicomputer-Based Information Retrieval System," Report No. UIUCDCS-R-76-779, M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, Illinois, February 1976.

MORT76 Morgan, T.J., "A Thesaurus Feature for the EUREKA Information Retrieval System, M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, Illinois, August 1976.

OSIN74 Osin, Luis, "A System to Produce CAI Courses from Logically Structured Educational Material," Technical Report No. 29 (November 1973), Doctoral Dissertation, Department of Computer Science, Israel Institute of Technology (Technion), Haifa, June 1974.

OSIN76 Osin, Luis, "SMITH: How to Produce CAI Courses Without Programming," International Journal of Man-Machine Studies, 8, 107-241, 1976.

RINE76 Rinewalt, J.R., "Evaluation of Selected Features of the EUREKA Full-Text Information Retrieval System," Report No. UIUCDCS-R-76-823, Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana, Illinois, September 1976.

STEL74 Stellhorn, W.H., "An Experimental Information Retrieval System," Report No. UIUCDCS-R-74-657, Department of Computer Science, University of Illinois, Urbana, Illinois, 1974.

4. IMPLEMENTATION DETAILS OF EURECAI MODULES

In this chapter we shall look into the implementation of the three CAI programs that have been developed. AUTHOR is a stand-alone program that creates the course description files (i.e., CRS.CAI). STUDNT is the program that creates student activity files (e.g., MORY.ACT). GUIDE, is the instructional program, which is a simplified version of SMITH's GUIDE. Finally, the essential modification of EUREKA modules are described.

4.1 AUTHOR

4.1.1 Purpose

The purpose of AUTHOR is to allow the author or the system programmers to create course description files.

4.1.2 Input

The input to AUTHOR is DBF.CAI, which is a contiguous file in ASCII form. At the present time there is a PL/1 program (called DBFOUT.CAI) which accepts input (author specifications) and prints two copies of DBF. One copy is clearly labeled and designed for the author or system programmer to verify the input. The second copy (which immediately follows the first copy on the printout) is DBF.CAI. Thus this part of output, at the present time, is extracted of

the output (using the EDIT-11) and labeled DBF.CAI. The contents of this file are the following variables and arrays (in the order of appearance):

NF: number of frames in the course.

NT: number of topics.

NUMEX: number of exercises (recall that exercises are considered as special frames).

TNRG: total number of (distinct) remedial groups. Recall that for every wrong answer of an exercise, the author would provide a list of topics to be reviewed. Each such list is a remedial groups (of topics). Different exercises may have identical remedial groups.

TNANS: total number of answers (of all exercises).

MAXL: maximum level for frames and students.

FT: frame vs. topic array of characters. It has NF x NT entries. Each entry is a character from the list: E, N, C, I, R or blank.

FRALIN: frame library numbers. This is a two-dimensional array (FRALIN (1:2, 1:NF)). For each frame, two numbers, corresponding to the document number and a paragraph number, are provided. This pair uniquely identifies a frame within a given data-base on EUREKA.

TOPIC: list (array) of topic names. This is a vector, with TOPLEN (topic length) characters per entry. TOPLEN is provided as an input to the PL/1 program (DBFOUT.CAI) and should be as long as the longest topic name.

GRADKY: This array has NF entries. Each non-zero entry corresponds to an exercise-frame and the entry is the exercise number.

SFL: sequence frame levels. There are (NF+1) entries in that copy of SFL that appears in DBFOUT.CAI, and used by GUIDE. Each entry corresponds to the level of the frame in the sequence. For example, SFL (5) = 2, means the fifth frame in the sequence is a level-2 frame. The last entry, SFL(NF+1), is always zero.

NANS: number of answers. NANS array has NUMEX entries. NANS(5)=2 means the fifth exercise has two answers (correct and incorrect). Only one correct answer is allowed per exercise.

FSTANS: first answer. FSTANS has NUMEX entries. Each entry points to the remedial group (in CORTOP) corresponding to the first answer of a given exercise. For example, FSTANS(5)=16, means that the remedial group number corresponding to the answer #1 of exercise number five is found at the sixteenth entry of CORTOP.

CORTOP: This array has TNANS entries, i.e., one remedial group per answer. The entries corresponding to correct answers are zero. All other entries contain a remedial group number. For example, FSTANS(5)=16, CORTOP(16)=4 means that the fourth remedial group corresponds to the first answer of the fifth exercise. Note that CORTOP (17) will correspond to the second answer of the fifth exercise!

TOPFAIL: topics failed. This two dimensional array provides a list of topics per remedial group, i.e., TOPFAL (1:TNRG, 1:NT). All entries are integers zero or one. For example, if TOPFAL(5,2:NT)=0 and TOPFAL(5,1)=1, then we have only one topic (topic number one) in the fifth remedial group.

FP: frame precedence array. This is a two dimensional array of the form FP(1:NF, 1:NF). Entries are zero or one. FP(50,100)=1 means frame number fifty must precede frame number 100 in the sequence of presentation.

ACTKEY: activity key array. This vector has entries with zero or one as value. Entries of one correspond to the exercise frames. For example, ACTKEY(7)=0, ACTKEY(8)=1 means seventh frame is expository and the eighth an exercise frame. This array is copied to every student's activity file (in STUDNT) and is updated as the student is presented with various frames.

All numeric entries in DBF.CAI are in I(5) format, with the exception of: TOPFAL, SFL, FP and ACTKEY which are in I(1) format.

4.1.3 Output

The output file generated by AUTHOR, is CRS.CAI. This file is contiguous and should be allocated before running AUTHOR. This is a binary file (not ASCII) for easy manipulation by GUIDE.

CRS.CAI has a heading with the format described in table 3.

All entries in the header are in entry per word format. All entries in the arrays are in entry/byte format with the exception described below. FP as expected is a binary array. TOPFAL is implemented as a bit structure. TOPFAL pointer (in the header) points to a table with TNRG pointers. Each pointer points to a list, with the first (byte) entry specifying the number of topic numbers in that list, followed by the topic numbers. FRALIN entries are in entry/word format (since document numbers > 255 are possible in EUREKA data-bases).

All array pointers are displacements with respect to the beginning of the buffer. They become physical pointers at run time.

4.1.4 Program Description

The AUTHOR program serves to reformat the course information into a suitable form (i.e., easy and fast access by GUIDE, and much more compactness). Two statically allocated buffers are used, one to read in DBF.CAI and one to write out CRS.CAI.

Table 3 CRS.CAI header

NF	
NT	
NUMEX	
TRNG	
TNANS	
MAXL	
FT	pointers to
FRALIN	the arrays
TOPIC	
GRADKY	
SFL	
NANS	
FSTANS	
CORTOP	
TOPFAL	
FP	
ACTKEY	
FT entries	
.	
.	
FRALIN entries	
.	
.	
etc.	

For more description of the programs, the reader is referred to the listing or the source code (AUTHOR.LST, AUTHOR.CAI, respectively).

This program runs under DOS (Disk Operating System).

4.1.5 Mode of Utilization

To run the AUTHOR program, a user logs in with the user code (77,6) and runs the program (i.e., types "R AUTHOR"). Error messages are described in 4.1.6. The proper response by the program will be: "CRSFIL IS CREATED".

4.1.6 Error Messages

"INPUT BUF IS SMALL": not enough space has been allocated to read DBF.CAI. Therefore, "CRSBLS" of CAIMAC macro in SYSMAC.SML should be increased accordingly. Size of the input buffer (in blocks) is calculated by multiplying the (CRSBLS" by four. There are 256 words per block. For example, if DBF.CAI is 40 blocks, then "CRLBLS" should be set to a value over 10. CRS.CAI must also be re-allocated with the new CRSBLS size.

"OUTPUT BUF IS SMALL": not enough space allocated for CRS.CAI buffer. "CRSBLS" is set to a value less than the size of CRS.CAI. It should be set to the same value or to a slightly larger value. "CRSBLS", as always should be changed in CAIMAC macro of SYSMAC.SML.

4.1.7 Notes

1. All the CAI files are under user number (77,6).

2. To verify correctness of CRS.CAI, a memory dump of this file should be obtained using the routine called EDUMP.

3. Every time "CRSBLS" or any other value in CAIMAC is changed all three CAI programs (i.e., AUTHOR, STUDNT and GUIDE) must be re-assembled, re-linked, etc.

4.2 STUDNT

4.2.1 Purpose

The purpose of this program is to create student activity files.

4.2.2 Input

CRS.CAI described in section 4.1 is the only input file to this program. There is also input provided by the user in interactive mode. This is discussed in section 4.2.5.

4.2.3 Output

The output of this program is a set of student files, named in the following format: NAME.ACT where NAME is the student's name which must be unique and have no more than six characters. As implied above, more than one student file may be created at any session.

The contents of the activity file is shown in table 4.

The description of the variable and arrays follows.

FILSIZ: the size of the activity file in bytes.

NAME1 & NAME2: six characters are packed in the RAD50 format into the two words. This is the student's name.

IOBUF: points to a temporary I/O buffer used during execution of GUIDE.

LKBKPR: Link block pointer. This points to the second word (higher address) of LKBKWDs. Link block addresses are kept in student's activity file to allow re-entrancy.

LKBKWD: Link block words. These two words are filled in immediately after the activity file is opened. The second word (higher address) is the address or pointer that DOS (Disk Operating System) provides upon opening the file. The first word is designated normally as a pointer to an error-handling routine, which the code does not use. Its presence, however is essential when using the Interpreter (INT45), for debugging. I suggest that we never remove such debug facilities from the code, even though we might think the code is super bugless!!

FTSCPR: full-text searcher pointer. This points to the AXBUFF.

IS: this is the internal sequence number of the current frame (i.e., the frame being presented). For example, when IS=5, the fifth frame is presented.

STEP: Every presentation (of a frame) increments the STEP. Note that, for example, any time the fifth frame is presented, IS=5, but STEP has a different value at each presentation.

INISTP: initial step. At the beginning of every session INISTP is set to the current value of STEP. Wherever the difference between these two values, STEP and INISTP, is equal to the value of SAVEFR, the activity file is stored for precautionary purposes and INISTP is set equal to STEP. SAVEFR is the frequency of saving (storing) the activity file. SAVEFR is currently set to ten.

Table 4. Student activity file header

FILSIZ			
NAME1			
NAME2			
IOBUF			
			Continuation
LKBKPR			
LKBKWDs		NF	ACTKEY entries
FTSCPR		(NF/8)	REVLST entries
IS			AXBUFF
STEP			Document #
INISTP			Par. #
STULEV			
TMPLEV			
ANSWER			
STA			
LOGPTR			
STKBEG			
STKPTR			
ACTKEY			
REVLST			
PATH			
PATH entries	PATSIZ		
STACK area	STKSIZ		

contd.

STULEV: student's level. Zero, one, etc.

TMPLEV: student's temporary level, used when student requests clarification.

ANSWER: student's last response is stored here.

STA: the state of his progress. His state of progress is normally "U" state.

LOGPTR: pointer to the logon block created by USRNTF module of EUREKA. This is used to perform I/O using the EXECUTIVE.

STKBEG: points to the beginning of stack. The size of this stack is determined by value of STKSIZ (in CAIMAC). It is presently forty bytes, which allows ten levels of storage (four bytes per level). STKSIZ must be a multiple of four.

STKPTR: points to the top of stack. The stack is used to remember the student's history of progress. The stack is primarily used when the student diverges from the main sequence of the frames (via a "C", "B" or a review). The information stored on stack are the current values of: STA, TMPLEV, IS and STEP.

ACTKEY: as defined in section 4.1.2.

REVLST: review list. It maintains the list of frames to be reviewed. This is a binary array, with one entry corresponding to a frame to be represented.

PATH: this array maintains the path of student's progress. The entries are the value of IS at different STEPs. The size of this array is determined by PATSIZ (in CAIMAC), currently set to twenty, which means the last twenty steps of the student's activity can be recalled.

AXBUFF: a buffer required for invoking the FTSRCH module of EUREKA, which given the document and paragraph number (as determined by FRALIN entries) displays the frame. Explanation of the entries in this buffer is left to EUREKA documentations! (You have the author's best wishes!!)

4.2.4 Program Description

The STUDNT program by using some information provided by the CRS.CAI file and the author (or any one running the program) creates student files. Activity files are dynamically allocated and are contiguous. Their size is a function of NF. For more details refer to the file descriptions (sections 4.2.2 and 4.2.3) and the source listing (STUDNT.LST).

This program is run under DOS (Disk Operating System)

4.2.5 Mode of Utilization

Figure 1 should demonstrate how to create student files.

4.2.6 Error Messages

If input or output buffers are too small, the user is informed accordingly. CRSBLS or ACTBLS should be increased accordingly. They are found in CAIMAC macro of SYSMAC.SML.

4.2.7 Notes

1. As always, if any of the parameters such as ACTBLS, STKSIZ, PATSIZ, etc., need to be changed, the change must take place in the CAIMAC macro found in SYSMAC.SML. Then, all the CAI modules (i.e., AUTHOR, STUDNT, and GUIDE) must be reassembled and re-linked, etc. All CAI modules may be found under (77,6) user number.

Figure 1

Note: All user entries follow the colon and are followed by RETURN key.

```
.RU STUDNT

PLEASE TYPE IN STUDENT'S NAME (MAX OF 6 CHARS) : LUIS

PLEASE TYPE IN STUDENT'S LEVEL : ø
ACTIVITY FILE IS NOW CREATED.

PLEASE TYPE IN STUDENT'S NAME (MAX OF 6 CHARS) : LONGNAME

PLEASE TYPE IN STUDENT'S NAME (MAX OF 6 CHARS) : LUIS

A FILE WITH THIS NAME ALREADY EXISTS!!!

PLEASE TYPE IN STUDENT'S NAME (MAX OF 6 CHARS) : OSIN

PLEASE TYPE IN STUDENT'S LEVEL : 3

LEVEL, GREATER THAN MAXLEVEL!!

PLEASE TYPE IN STUDENT'S LEVEL : 2
ACTIVITY FILE IS NOW CREATED.

PLEASE TYPE IN STUDENT'S NAME (MAX OF 6 CHARS) :

THANK YOU.  I'LL SERVE YOU LATER
.
```

2. To verify correctness of the output files, a dump of a given file through EDUMP routine must be obtained, and compared with the file description described in section 4.2.4. CRS.CAI correctness should, however, be verified first.

3. Student files are so protected, to allow their updates from other user numbers (under which GUIDE and EUREKA operate).

4.3 GUIDE

4.3.1 Purpose

The purpose of this program is to provide instruction and interact with the students in a mixed-initiation CAI mode.

4.3.2 Input

There are two types of input files for this program. One is the course information file (i.e., CRS.CAI), and the other is a student activity file. At any given point the program has access to one file of each type. The format of these files was described in detail in sections 4.1.3 and 4.2.3.

The other input to the program is the student's response or request. This input is often only one character or one digit (in exercises).

4.3.3 Output

The only files being updated by this program are the

students' activity files. The system communicates with the student (through a CRT or teletype) in form of easy to understand messages. All the frames are displayed by FTSRCH module of EUREKA upon GUIDE's demand.

4.3.4 Program Description

This program as mentioned earlier, is an adopted (simplified) version of the original GUIDE module of SMITH. Therefore, in this section only those features different from that of the original GUIDE, and those unique to this implementation are pointed out.

The features that were not adopted in this implementation of GUIDE are listed below.

1. There are no frame tags or student tags.
2. There are no extensive frames.
3. There are no response analysis facilities for exercises. All answers are single digits corresponding to multiple choice exercises.
4. There are no comment frames.
5. Remedial loops are treated differently. In this implementation a wrong answer to an exercise (the first attempt) will result in review of all N (necessary) frames which are explicitly specified to precede the exercise. Note that if no frame satisfying both requirements is found, then the N-frames are presented. Such N-frames are available, since an exercise should not be testing a topic that has not been taught.

If it is the second attempt, a wrong answer will trigger presentation of all N and C-frames. Subsequent attempts will result in presentation of the same frames.

Some new features of this implementation which the author also recommends for other implementations are listed below.

1. The student is presented with descriptions of all the student commands at the start of his first session, and this help is also available to him upon his Help (H) command.
2. The answer to an exercise may be obtained by typing Answer (A). This, however, is not available to the student unless he has already made an unsuccessful attempt!
3. When the student wishes to review some topics, he may request to view list of ALL (A) topics, and the corresponding topic numbers.
4. Upon completion of the course, the student is given a chance to review any topic he wishes. This allows the student to return at any later date and review topics of interest (e.g., for an exam).
5. Size of various variables such as, length of topic texts (TOPLEN), size of the stack (STKSIZ), size of the path array (PATHSIZ), frequency of precautionary storage of activity file (SAVEFR), can be very easily changed, since all such variables are defined at only one place (i.e., in list of all CAI macros -- CAIMAC).

The program is written in MACRO-11 as are all other programs in EURECAI system. GUIDE is recognized as one of

the many tasks running under the executive system. Therefore, the linked code (at start address of 64000 octal), is placed in the library of EUREKA modules (via CLIB routine). GUIDE is invoked from the USRNTF module and it, in turn, invokes FTSRCH module. Some I/O commands do directly request service from DOS. This is partly because I/O in binary format is not implemented in EXECII, and mainly to allow code re-entrancy.

The program is, quite modestly, a very well structured and commented code. It has been designed to be modular, understandable, and easily modified. As much as possible, the choice of parameters and labels have been made similar to those, appearing in the original code of GUIDE (written in PL/1), for easier reference.

4.3.5 Mode of Utilization

This program is invoked by the student, after he has logged on to EUREKA, by typing "GUIDE".

4.3.6 Error Messages

There are no error messages concerning the code, system or files generated by GUIDE, that require any action by system programmers or author. If a student has no activity file (under the name he typed in), he is so informed.

All the messages generated by GUIDE are to guide the student concerning his mode of interaction.

4.4 EUREKA Modifications

Some EUREKA modules had to be modified to allow linkage of GUIDE with EUREKA.

The USRNTF module was modified to check for the "GUIDE" command and the presence of the correct data-base in the system. Upon receiving the command GUIDE, USRNTF invokes the task GUIDE. GUIDE does not return to the father task until the student suspends his lesson.

The FTSRCH module of EUREKA was also modified: 1) to print the paragraph and document number of the text being displayed in the BROWSE mode and 2) to display the paragraphs requested by GUIDE and return the control back to GUIDE.

5. A GUIDE FOR EURECAI USERS

In this chapter, some information that may be useful to authors of CAI courses and their students are presented.

5.1 On Creating a Course (for authors)

This section is provided to help non-computing authors interested in using EURECAI to create some CAI course. Therefore, let us formulate a suggested sequence of actions that he should take.

1. Read chapter three of this thesis (reference to (OSIN76) is highly recommended).

2. Read through section 4.3.4 to learn about the differences between the original system (described in section 3.1) and the current implementation.

3. Find out through Dr. D.J. Kuck (under whose supervision the system is maintained) if the materials of interest (the course contents) are available in EUREKA data-bases. If not, you should request to have the text from publications of your choice (i.e., the library of frames) entered into a data-base. Note that all the text of interest must be collected into one data-base.

4. Learn how to use EUREKA so you may search the topics of interest and display various documents in the data-base. This task is facilitated for you by offering you a CAI course that teaches you about EUREKA.

5. You are now ready to select and sequence the frames of interest from the text in EUREKA data-base. Every frame is identified by document number and paragraph numbers (these numbers are printed on top of each paragraph being viewed when in the BROWSE mode).

(a) Decide on the topics you would like for the course to cover. Make a list of the topics and number them (starting with topic number one with increments of one). Note that, having more topics in the course (i.e., less decomposable topics) may result in presentation of more relevant frames in the remedial loops. It also allows the student to review specific topics of interest, rather than navigating through the frames covering a more general topic.

(b) Obtain a printout of the documents related to these topics. You may have to use the BROWSE mode to label the paragraph boundaries on the printout of each document.

(c) Choose frames of interest and assign a level to each (recall the significance of frame levels). You should also specify the topics related to each topic and their qualifiers.

(d) If not already available in the text, you must design a generous number of exercises. In this case, you should write the text and enter each exercise as one paragraph into the EUREKA data-base.

At the present time, this feature is not available, but the system programmer, Perry Emrath, is in the process of providing software to allow updating of the data-base contents.

Therefore, seek out his documentation or contact the EURECAI system programmer.

(e) Decide upon the sequence of the presentation of the frames (including exercises) and label them accordingly (starting with one and increments of one). Use the criteria utilized by SEQUENCE of SMITH (section 3.1.2).

6. Recall that exercises are special frames. Remember that exercises of various difficulty, testing various topics play an important part in effectiveness of the instructional program. Make a list of all exercises (identified by the frame sequence number and the exercise number). For each incorrect answer to an exercise, provide a list of topics that should be reviewed. Now, extract from all these lists of topics, a redundant list of remedial groups. Each group is identified by its number.

7. Provide all precedence relationships. Note that this is useful in design of remedial loops for exercises. For example, let's assume a wrong answer to an exercise means reviewing a topic that has been described in, say, ten necessary frames, but only two of these frames are related to that exercise. Specifying those two frames as to precede the exercise will guarantee the presentation of only those two after the student's first unsuccessful attempt. This is a very powerful tool in providing helpful and relevant material to the student.

8. Request from EURECAI programmers to aide you with creating the course files. There is a PL/1 program, found

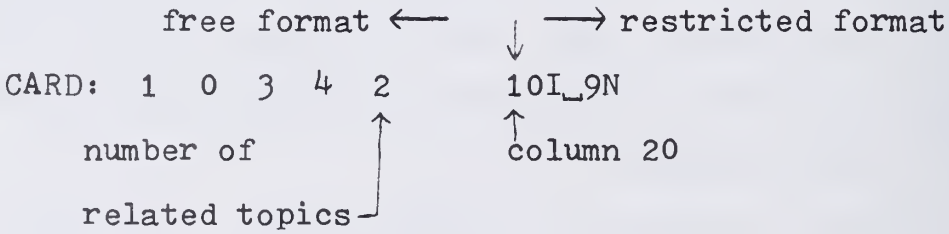
under DBFOUT.CAI (77,6) that accepts information about the course and creates the course description file. The program also provides an output for input verification.

The input format is as follows.

(a) The first input card contains two numbers (in free format) -- the number of topics and length of the topics text (the longest topic). TOPLEN used in EURECAI modules is currently set to 30 characters.

(b) List of topic names (text), in order. One topic per card, left justified.

(c) Frame description cards. The following example demonstrates the format:



This card describes: frame #1, level-0, document 3, paragraph 4; introduces topic 10, and is necessary for learning topic 9.

(d) Number of exercises in the course (one card).

(e) Exercise description cards. One card per exercise, with the following included: exercise #, frame #, # of answers followed by remedial group numbers corresponding to the answers. Example:

CARD: 1 5 3 2 0 4

describes exercise #1, frame #5, with three answers, where second answer is the correct one. The first and third have remedial groups #2 and 4, respectively.

(f) Remedial groups. One card per group. Example:

CARD: 2 3 9 10 1

describes remedial group #2, consisting of three topics: 9, 10, and 11.

(g) Frame precedence relations. Each relation is described by a pair. Entries must be ended by a pair of zeros on the last card. Example:

CARD: 1 7 3 9 0 0

frame # precedes 7, and 3 precedes 9.

9. The output of PL/1 program has two forms. One is for the use by the programmers, and the other to be used as an input to the AUTHOR program. This output should be placed in DBF.CAI (77,6), then the AUTHOR program should be run, which creates CRS.CAI (77,6), as described in section 4.1.5.

This concludes the course construction process.

10. The STUDNT program should now be run to create student files, as described in section 4.2.5. You should probably create a student file for yourself and test the instructional program, to further improve the course organization. If more improvements seem essential go back to step five.

11. Hooray! To this author's surprise (!) you have now successfully created the course and certainly deserve his congratulations.

In the process of testing the EURECAI system, a course was created to provide instructions in how to use EUREKA system. The time spent on steps five through seven was approximately five hours. The user's guide of EUREKA (MORG76) made up most of the course material. Therefore, the above figure includes providing the exercises and course specifications.

5.2 On Taking a Course (for students)

This section is designed to provide some information for those who intend to take courses implemented on EURECAI system.

GUIDE, which is the teaching program, has been designed to be self-explanatory through easy to understand messages; but knowing some about the teaching strategy and the student commands may be useful. Therefore, let us learn how to communicate with GUIDE.

In teaching the student, GUIDE presents(displays) a frame (a paragraph) of material. This could be some facts, examples or a exercise related to a concept. The student's response to a frame can be one of the following:

U (Understood): If you understand the material presented in the frame (or think you do) you should use this command. If you understand the concept but would like to see more explanation or examples you should still type U, since there could be such materials presented in the following frame.

C (Clarify): You use this command either if you do not understand the material (i.e., if the transition from the previous frame to this frame was not that smooth), or if you understood the previous frame but you expected to see more examples or explanation related to the previous frame before reading about the concepts introduced in the current frame. Note that there is not always additional material available to satisfy your request, in which case you will be informed accordingly.

B (Back): This command presents you with the previous frame. Successive B's take you back in the sequence you followed. The exercises, however, are not presented when going back in the sequence.

? (doubts about some topic, review mode): This command allows you to review any topic you have covered up to the current frame. Upon entering the review mode you may specify the topics you wish to review, by providing the topic numbers. To obtain topic numbers you may either request a list of all topics in the course, or the topics related to the current frame. Note that the latter is a nice way of finding out how a frame is related to various topics covered in the course. Upon ending your entries in the review mode GUIDE will present frames of material directly related to the topics you wished to review.

Note also that when you answer an exercise incorrectly the system will automatically present frames to review those

topics related to your mistake. If you have a lot of trouble with an exercise, the system will provide you a lot of material!

R (Resume): Any time you give the commands C, B or review a set of topics, you temporarily halt your forward progress in the main sequence of the frames in the course. In order to allow you to return to the main sequence, this command is made available to you. The best use of this is in exercises. After making a mistake in answering an exercise and beginning review of the corresponding topics, if you know what the correct answer is or do not wish to continue with the review, give the R command and return to the exercise.

S (Suspend): This command closes the session, while remembering the current frame, to be presented at the start of the next session.

A (Answer): If you have failed an exercise and did not find the review material helpful, you may request the correct answer to the exercise by typing A.

H (Help): This command provides you with a brief description of all the commands we've discussed here.

To get access to GUIDE at the terminal, you should first ask the system programmer to load the EURECAI system. When the system requests that you "logon", you type "LOGON" followed by a blank and your first name (6 characters or less). Now, you may enter GUIDE by simple typing GUIDE, and following instructions.

LIST OF REFERENCES

OSIN76 Osin, Luis, "SMITH: How to Produce CAI Courses Without Programming," International Journal of Man-Machine Studies, 8, 107-241, 1976.

6. FUTURE DEVELOPMENTS

In this section a list of suggestions to improve the present CAI system is presented.

An obvious observation would be that we could incorporate all the features of SMITH into our system. However, we shall refer only to those features and modifications that this author finds more essential and useful to our system.

6.1 The Student Program

At the present time, there are two user files for each student, one recognized by GUIDE and created by STUDNT and one recognized by EUREKA and created by the INTUSR program.

The STUDNT program should be modified so that it creates a EUREKA userfile automatically, when creating an activity file. This is a rather simple modification.

Furthermore, the activity files should be extended to include storage for student performance-related statistics.

6.2 The AUTHOR Program & Author Facilities

With a glance at the description of the process of course construction (section 5.1), the reader realizes that there is a moderate amount of room for facilitating this procedure.

The process of creating a course should involve only one program which:

(a) accepts author's specification in a reasonable format

(b) validates all the input specifications, detects all inconsistencies, recognizes the holes in the curriculum and reports its findings in an understandable form

(c) has the capability to sequence all the frames based on the information provided by the author

(d) provides a clearly labeled output of the course apecification to serve as an aid and reference to the author

(e) creates CRS.CAI file upon the complete and error-free specification.

This is probably a master level project due to complexities involved in the sequencing process (refer to OSIN74, section 2.5). The PASCAL compiler available on the PDP-11 system should be very helpful in the required programming.

A partial implementation of this program (one that does not satisfy all of the above requirements) may still prove helpful.

6.3 The Instructional Program, GUIDE

One of the shortcomings of GUIDE, is the absence of student response analysis. All questions in exercises are multiple choice. Therefore, facilities should be provided to allow fill in the blanks or more complicated questions. This may dictate maintaining a directory of equivalent words and / or detecting spelling errors.

Addition of comments frames may result in more effective handling of the student's weaknesses in the exercises.

GUIDE should also be modified to collect various performance-related statistics on the students such as: amount of time spent on frames, number of attempts on exercises, pattern of clarification requests, etc. As pointed out earlier this also implies extending the activity file.

In this author's opinion, the remedial loops of exercises should not only consist of related necessary frames but also should include unpresented exercises (of higher level, thus providing more "drill and practice" in the areas most needed by the student. For the same reason, exercises should be included in the remedial loops generated by Clarify command, even when they are the only frames available between the last understood frame and the correct one. This will allow the student, access to more exercises as he feels necessary (a desirable characteristic of generative and "drill and practice" CAI systems).

An alternative approach would be to present a comment frame associated with the correct answer of an exercise, where the comment frame has a sequence of extension frames, each providing an example or a solved exercise. The student may follow as many extension frames as he finds necessary.

6.4 The EURECAI System

Facilities must be provided to allow a non-programmer author to attach frames of text such as exercises to EUREKA

data-bases.

A communication scheme is available to allow sharing common data in a contex-switching environment. This scheme should be used to allow presence of only one copy of CRS.CAI in the memory.

LIST OF REFERENCES

OSIN74 Osin, Luis, "A System to Produce CAI Courses from Logically Structured Educational Material," Technical Report No. 29 (November 1973), Doctoral Dissertation, Department of Computer Science, Israel Institute of Technology (Technion), Haifa, June 1974.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-77-902	2.	3. Recipient's Accession No.

4. Title and Subtitle

IMPLEMENTATION OF AN
INFORMATION-RETRIEVAL BASED CAI SYSTEM

5. Report Date: August 1977

6.

7. Author(s): Morteza Bahar

8. Performing Organization Rept. No. UIUCDCS-R-77-902

9. Performing Organization Name and Address

University of Illinois at Urbana-Champaign
Department of Computer Science
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.: US NSF MCS73-07980

12. Sponsoring Organization Name and Address

National Science Foundation
Washington, D. C.

13. Type of Report & Period Covered: Master's Thesis

14.

15. Supplementary Notes

16. Abstracts

As a simplified version of GUIDE, the instructional program of SMITH CAI system was adopted and implemented in connection with EUREKA -- an experimental information retrieval system.

The thesis also includes a discussion on the design of CAI systems, a brief description of SMITH and guidelines for the authors and students of the EURECAI system.

17. Key Words and Document Analysis. 17a. Descriptors

Computer-Aided Instruction
Course Creation
EUREKA
Information Retrieval
SMITH
Text Processing

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	UNCLASSIFIED	78
	20. Security Class (This Page) UNCLASSIFIED	22. Price